AFRL-RI-RS-TR-2014-277

DATA STREAM MINING BASED DYNAMIC LINK ANOMALY ANALYSIS USING PAIRED SLIDING TIME WINDOW DATA

NOVEMBER 2014

FINAL TECHNICAL REPORT

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE

■ **AIR FORCE MATERIEL COMMAND** ■ **UNITED STATES AIR FORCE** ■ **ROME, NY 13441**

NOTICE AND SIGNATURE PAGE

REPORT DOCUMENTATION PAGE	*Form Approved* **OMB No. 0704-0188**

1. REPORT DATE *(DD-MM-YYYY)* NOVEMBER 2014	**2. REPORT TYPE** FINAL TECHNICAL REPORT	**3. DATES COVERED** *(From - To)* APR 2011 – APR 2014

4. TITLE AND SUBTITLE DATA STREAM MINING BASED DYNAMIC LINK ANOMALY ANALYSIS USING PAIRED SLIDING TIME WINDOW DATA	**5a. CONTRACT NUMBER** IN-HOUSE
	5b. GRANT NUMBER N/A
	5c. PROGRAM ELEMENT NUMBER 61102F
6. AUTHOR(S) Keesook Han (AFRL), Tao Zhang (Central Michigan University), and Qi Liao ((Central Michigan University)	**5d. PROJECT NUMBER** GGIH
	5e. TASK NUMBER ZO
	5f. WORK UNIT NUMBER RR
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Air Force Research Laboratory/RIGA 525 Brooks Road Rome NY 13441-4505	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Air Force Research Laboratory/RIGA 525 Brooks Road Rome NY 13441-4505	**10. SPONSOR/MONITOR'S ACRONYM(S)** AFRL/RI
	11. SPONSOR/MONITOR'S REPORT NUMBER AFRL-RI-RS-TR-2014-277

12. DISTRIBUTION AVAILABILITY STATEMENT

Approved for Public Release; Distribution Unlimited. PA# 88ABW-2014-5054
Date Cleared: 31 OCT 2014

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Dynamic network analysis for network security is challenging because it is computationally expensive to extract knowledge structures for quantifying the security levels of dynamic networks. There has been an increased interest in dynamic network analysis for network security and it is an emergent scientific field in network science. In this report, we introduce network analytics metrics and sliding time window data structures for data stream mining in order to incorporate link anomaly detection into the dynamic network analysis. The proposed dynamic link anomaly detection framework provides the capability to construct effective knowledge structures by measuring the security levels of dynamic networks, and filtering anomalous or suspicious links from network flow data. In addition, the sliding time window based method produces useful processed stream data for generalized dynamic network analysis.

15. SUBJECT TERMS

Data Stream Mining, Sliding Time Window, Network Security, Dynamic Network Analysis, Dynamic Link Anomaly Analysis, Dynamic Link Anomaly Detection, Analytics Metrics, Node-Dependent Metric, Path-Dependent Metric, Link Appearance Metric, Security Levels, Multiclass Classification, Network Characterization

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** KEESOOK J. HAN
a. REPORT U	**b. ABSTRACT** U	**c. THIS PAGE** U	UU	35	**19b. TELEPHONE NUMBER (*Include area code*)** N/A

TABLE OF CONTENTS

LIST OF FIGURES iii

LIST OF TABLES iii

1 SUMMARY 1

2 INTRODUCTION 2

3 METHODS, ASSUMPTIONS, AND PROCEDURES 4

3.1 Network Analytics Metrics 4

3.1.1 Node-Dependent Metric 4

3.1.2 Path-Dependent Metric 5

3.1.3 Link Appearance Metric 5

3.2 Dynamic Link Anomaly Detection 6

3.2.1 Overview of the Framework 6

3.2.2 Sliding Time Window Algorithm 11

3.2.3 Link Scoring Algorithm 12

3.2.4 Link anomaly detection Algorithm 13

4 RESULTS AND DISCUSSION 14

4.1 Case Study 14

4.1.1 Data Sets and Procedures 14

4.1.2 Sliding Time Window Size Selection 14

4.1.3 Thresholds for Cyber Attack Detection 16

4.2 Performance Evaluation 19

4.2.1 Link Anomaly Classifications 19

4.2.2 Multi-Thresholds Selection 22

4.2.3 Performance of Classification 22

5 CONCLUSION 25

6 REFERENCES 25

INDEX 28

LIST OF FIGURES

1 **Data Flow Diagram** 7

2 **Parameter Setting** 15

3 **Link Appearance Indexes of an IP-Port Graph** $\omega = 1min$ 15

4 **Link Appearance Indexes of an IP-Port Graph** $\omega = 15min$ 16

5 **Link Score Distribution of an IP Graph and** θ_{build} 17

6 **Link Score Tables** 18

7 **Link Score Distribution of an IP Graph, Logs and Anomaly List** . . 18

8 **Multiclass Tree Representation of** 4×4 **Confusion Matrix** 19

9 **Hierarchical Aggregated Functions of Confusion Matrices** 20

10 **3-Threshold Selection Procedure Using a Binary Classifier** 22

11 **Alternative Comparison of Various Thresholds in ROC Curve** . . . 23

12 **Performance Evaluation Results** 24

LIST OF TABLES

1 **Link Classes and Mathematical Notations** 12

2 **Terminology and Derivations of Classification Functions from Confusion Matrix** 21

1. SUMMARY

Network security is becoming increasingly challenging because of complexity and dynamics of network interactions. Dynamic network analysis (DNA) is an emergent scientific field in network science. There has been an increased interest in data stream mining based DNA for network security. The major challenge of this dynamic network analysis is that it is computationally expensive to extract knowledge structures for measuring the security levels of large scale dynamic networks. This research aims to develop novel technical methods in order to solve this challenge. This report presents practical methodologies of data stream mining based dynamic link anomaly analysis (DLAA) using novel sliding time window structures and network analytics metrics.

The DNA is fundamentally tied to spatio-temporal analysis and the proposed DLAA is the specialized network security research area of the DNA. We employ spatio-temporal link analysis using paired data such as prior and current window data sets. The major purpose of applying the sliding time window algorithm is to process manageable and meaningful paired data to perform unsupervised learning. Each of the paired data sets is comprised of two types of data such as spatio-temporal data and spatial data. Spatial data structure can be modeled as a graph to represent spatial data that is a subset of spatio-temporal data. For the DLAD, unsupervised learning is appropriate to learn and measure the past histories of node-dependent, path- dependent and link appearance behaviors of dynamic networks from the prior window data and predict link anomalies in the current window data.

For the DLAA, we have developed a dynamic link anomaly detection (DLAD) framework to quantify the security levels of dynamic networks. The DLAD framework consists of three algorithmic components including sliding time window, link scoring and link anomaly detection algorithms. These components are operated by flow-based programming procedures. The proposed framework was tested and evaluated using the VAST 2011 Mini Challenge 2 (MC2) datasets, including a case study. This case study was implemented with the IP graphs and IP-port graphs, and illustrates the data analysis techniques. The performance of the proposed framework is fully described using various statistical measures (i.e., sensitivity, specificity, accuracy, precision and error rate, etc.) of the performance of a binary classification test. Through performance evaluation in the case study, we demonstrate that the proposed framework has the capability to construct effective knowledge structures for measuring the security levels of large scale dynamic networks, and to produce effective processed stream data for generalized DNA.

2. INTRODUCTION

The analytic computational complexity of networks has become increasingly challenging over the past decades due to the ever growing size of networks and the explosion of information exchanged over the Internet. Large scale networks can suffer from threats in many ways. For example, network anomalies such as failure events of IP backbone, Denial of Service (DoS) attacks, worm propagation, and network equipment outages have distinct characteristics but are all potential threats, which may strongly affect the functionality and dependability of networks in varying degrees.

In areas of network security management, network intrusion and anomaly detection [7, 13] can be roughly categorized as signature-based or data mining-based. Signature-based schemes have the advantage of low false positives if signatures are well defined patterns in advance. However, these schemes are not appropriate to detect zero-day attacks. There has been progress in using network behavior anomaly detection (NBAD) to detect or predict unknown anomalies. As an integral part of network behavior analysis, the NBAD still needs to establish knowledge of legitimate network connectivity and user behaviors over a certain period of time. Machine learning based anomaly detection may be promising because predefined signatures are not required, but challenges remaining to researchers are lack of recent attack data and clean (attack-free) training data, difficulty of evaluation, and high cost of errors [22]. There have been anomaly detection works looking at various aspects of network anomalies. We note that the difference and focus of this work is on detecting link anomalies in dynamic graphs that exhibit on/off patterns based only on network topologies in the network security management domain.

Network anomalies have been manually or automatically identified in network traffic data by applying the subspace method using multivariate time series of byte counts, packet counts, and IP-flow counts [13]. Neighborhood formation has been studied for bipartite graphs [23]. Leman Akoglu et al. [1] proposed the OddBall algorithm to find anomalous nodes in a graph. The authors use *egonets*, that is, the induced subgraph of an individual node (*ego*) of interest and its neighbors, and assign an outlierness score to an individual node. Power law density, weights, ranks and eigenvalues of the neighborhood subgraphs have proved useful for anomaly detection. The graph-theoretic approach [11] has also been applied to detect anomalies in email networks from the publicly available Enron email corpus.

In data mining domains, link mining [4, 8] and link prediction algorithms [17, 18] have been introduced. In particular, there has been recent progress in link prediction and its applications for a variety of domains such as social networks [10, 15], co-authorship networks [19], healthcare and bioinformatics [2]. However, most works in link prediction are only interested in predicting whether a pair of nodes that are not connected previously will ever be connected in the future. Although link prediction algorithms will predict the feature connections in a graph, most of them use a static graph as learning set without considering inherent temporal information from the network traffic data. Often, the network presents various dynamic behaviors [20] and traditional link prediction does not work well on datasets with temporality. Instead, we focus on the task of detecting link anomalies [1, 11, 21, 23, 24] and address the dynamic ‘on/off” behavior of link anomalies, e.g., whether and when a previously connected link will become disconnected. The malicious activities, which we refer to as anomalies, may be related to faulty hardware/software, misconfiguration, or security related events caused by malicious users and applications. While there have been some efforts in anomaly-based intrusion detection [7], the detection of link anomalies has remained challenging.

In this report, we introduce the network analytics metrics and sliding time window data structures for the data stream mining in order to incorporate the link anomaly detection into the DNA. Using knowledge structures from sliding time window data and network analytics measures, we study how to measure and analyze network security on topological structures. For better defense against zero-day attacks, we aim to utilize behavioral or learning based approaches, such as a link prediction algorithm based approach which analyzes network connections. The challenge lies in the fact that there are characteristics of anomaly detection fundamentally different from link prediction tasks [18]. Although link prediction [10, 15] may predict that a pair of nodes not connected in the past will be connected sometime in the future, it does not consider pairs of nodes that have previously been connected. The fundamental deficiency of traditional link predictions is that they do not consider the dynamics of network connections, e.g., the on/off patterns. It has been observed that networks are not only becoming much larger but much more heterogeneous, complex and dynamic as well. In computer networks, the massive amount of traffic among the computing nodes is constantly changing. For example, users and/or applications may establish and disconnect the connections at any time. Different types of anomalies are formally modeled by considering every possible combination of previously unlinked/linked nodes and currently unlinked/linked nodes. In each sliding time window, each pair of nodes is assigned a link score to measure its importance. The link scores will be used to predict whether the connections will be established or disconnected in the next time phase for a variety of situations. Based on the actual connectivity in a current snapshot graph, we are able to judge whether each link is anomalous or normal by examining the difference between the expected result and reality. Through a case study and performance evaluation on publicly available datasets, we illustrate the effectiveness of the link anomaly detection algorithm by comparing the evaluation metrics. We conclude that while this research has immediate benefits for network security management in terms of security investigation and troubleshooting, the proposed methodologies are general enough to have potential impact on many other types of network applications. Time-efficient security-related investigation and effective troubleshooting can be achieved by using the sliding time window based DLAA. For instance, network managers and administrators are able to monitor manageable amounts of latest link anomalies for further security investigations and threat preventions.

The main contribution of this paper is the development of the dynamic link anomaly detection (DLAD) framework. Three algorithmic components of the framework are sliding time window, link scoring and link anomaly detection algorithms. To keep the algorithms generic, our approach does not need any background information such as node attributes, but it is purely based on network topologies. Intuitively, the frequency of link appearances is a critical factor to classify link anomalies. It is reasonable to assume that the connections occurring close to the time of investigation may receive more emphasis than connections happening much earlier due to the temporal locality of packet exchanges and network flows. Hence, our approaches to link analysis employs temporal dynamic, similarity and centrality measurements on evolving network topologies in consecutive sliding time windows. These temporal dynamic, and similarity and centrality factors are combined to compute an overall link score as a coherent link anomaly measurement methodology.

3. METHODS, ASSUMPTIONS, AND PROCEDURES

3.1. Network Analytics Metrics

For various network analysis including social network analysis, node-dependent and path-dependent metrics are generally used to characterize topological structures [18]. However, these measurements are insufficient to characterize the dynamic network behaviors. Therefore, we introduce network analytics metrics for spatio-temporal link analysis approach to score security levels of dynamic links using paired sliding time window data.

In this section, we begin by introducing the network analytics metrics using graphs. A graph is an ordered pair $G = (V, E)$ comprising a set V of vertices or nodes $u, v \in V$ together with a set E of edges or links $(u, v) \in E$. Let $G = (V, E)$ be an unweighted graph that represents the topological structure of a general connected network. For each $u \in V$, let $\Gamma(u) = \{v : (u, v) \in E\}$, the set of vertices connected to u. In addition, for each $l \leq 1$, $paths_{u,v}^{(l)}$ is the set of paths from u to v consisting of l edges (path of length l).

3.1.1. Node-Dependent Metric

The Jaccard index, also known as the Jaccard similarity coefficient, was originally designed to measure the similarity between sample sets. For this report, the Jaccard index is selected as a *node-dependent metric* to characterize nodes' community similarity. It is defined as the size of the intersection divided by the size of the union of the sets [5]. In the graph G, for each pair of nodes $v, u \in V$, the Jaccard index is defined as the ratio between the number of their common neighbors and the number of total neighbors, namely:

$$J(u, v) = \frac{|\Gamma(u) \cap \Gamma(v)|}{|\Gamma(u) \cup \Gamma(v)|} \tag{1}$$

where, as previously, the cardinality of a set is denoted by placing absolute value signs around the set.

In our approach, $J(u, v)$ indicates whether network nodes $u, v \in V$, such as clients, servers, and routers, have a large percentage of overlapping destinations regardless of the absolute number of connecting targets.

3.1.2. Path-Dependent Metric

The Katz index is useful in measuring centrality of which determines the relative importance of a node within the network. The Katz index is selected as a path-dependent metric in order to characterize centrality of the nodes in a graph. Originally L. Katz [12] proposed a path-ensemble based proximity measure. The unweighted Katz index is used to measure the centrality of a node by measuring a variant of the shortest-path [5], with two nodes to be strongly related if their connecting paths tend to be short in length. The Katz index is defined as [16]

$$K(u,v) = \sum_{l=1}^{\infty} \beta^l \cdot |paths_{u,v}^{(l)}|, u, v \in V, \tag{2}$$

where $0 < \beta < 1$ is an attenuation parameter [3] ensuring that shorter paths contribute more to the score [16].

In our approach, $K(u,v)$ is computed to quantify link connectivity of nodes u and v.

3.1.3. Link Appearance Metric

The nonlinear weight function is used for the *link appearance metric* that characterizes the temporal dynamics of connected nodes for link anomaly detection [14].

The nonlinear weight function is defined as

$$w\left(t\right) = e^{-\lambda\left(1-\dfrac{t}{T}\right)}, t = 1, 2, ..., T \tag{3}$$

where λ is an attenuation parameter (i.e., positive real number), and t is the t^{th} snapshot graph and T denotes the total number of snapshot graphs in the prior window.

The Link Appearance Index $L(u,v)$ for $(u,v) \in E$ is defined as

$$L(u,v) = \frac{\sum_{t=1}^{T} w\left(t\right) \cdot \alpha_t}{\sum_{t=1}^{T} w\left(t\right)}, \tag{4}$$

where $\alpha_t \in \{0,1\}$ represents the link appearance at the t^{th} snapshot in the prior window. That is, $\alpha_t = 1$ is if the link is present (u, v connected) and $\alpha_t = 0$ if the link is absent (u, v disconnected).

Equation 4 yields a weighted mean, taking into account the fact that appearances of links at later snapshot graphs (or in other words closer to the time of investigation) should have higher weights over the earlier graphs. It is reasonable to make such an assumption due to the inherent temporal locality of network connections. Schemes such as network caching and

Cisco NetFlow also assume that if a packet is observed between a source-destination IP/port pair, it is likely that the source will send packets again to the destination in the near future. For temporal link analysis, higher weights are assigned for later link appearances because later appearance of connections is more important than earlier ones.

The *Node-dependent metric* (i.e., Jaccard index, Equation 1) and the *path-dependent metric* (i.e., Katz index, Equation 2) are used to measure node similarity and link centrality. These metrics are insufficient to detect anomalous links in dynamic and complex networks. Moreover, the complexity of computational time and memory are crucial factors to implement real applications, but the Katz index has a limitation when applied to real-world networks because its cubic time complexity $O(|V|^3)$, where $|V|$ is the number of vertices in V[3, 9], which makes it infeasible for analyzing a large network. In order to solve the fundamental cubic complexity problem of the Katz measure and make the calculation more efficient for large networks, we apply a heuristic method of partial simple paths by using the maximum length as a limiting factor. Paths which are longer than the maximum value will not be counted, partly due to the fact that in most cases, paths of shorter lengths will dominate the measure. In addition to these metrics, the *link appearance metric* is also used to implement feasible and efficient link anomaly detection mechanisms.

3.2. Dynamic Link Anomaly Detection

This section describes the overview of the dynamic link anomaly detection framework, and introduces sliding time window, link scoring and anomaly detection algorithms.

3.2.1. Overview of the Framework

Figure 1 shows data flow diagram of overall processes of a component-based framework for the link anomaly detection. Major algorithmic components of the framework consist of three algorithmic components that are sliding time window, link scoring, and link anomaly detection algorithms. These components are illustrated in Figure 1 and their algorithms are introduced in this section.

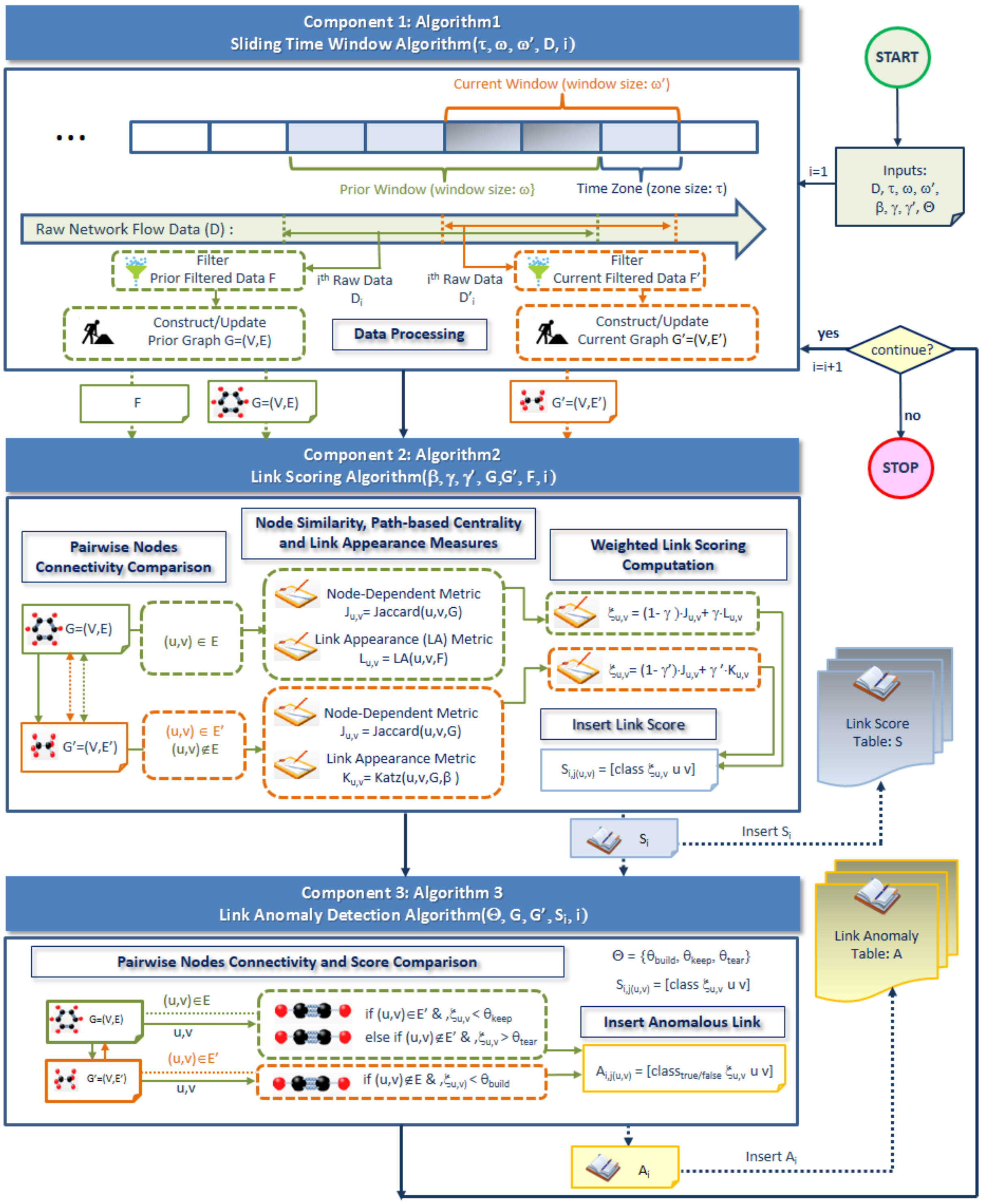

Component 1: Algorithm1
Sliding Time Window Algorithm(τ, ω, ω′, D, i)
Current Window (window size: ω′)
Prior Window (window size: ω)
Time Zone (zone size: τ)
Raw Network Flow Data (D) :
Filter
Prior Filtered Data F
ith Raw Data Di
ith Raw Data D′i
Filter
Current Filtered Data F′
Construct/Update
Prior Graph G=(V,E)
Data Processing
Construct/Update
Current Graph G′=(V,E′)
START
Inputs: D, τ, ω, ω′, β, γ, γ′, Θ
i=1
yes
continue?
i=i+1
no
STOP
F
G=(V,E)
G′=(V,E′)
Component 2: Algorithm2
Link Scoring Algorithm(β, γ, γ′, G,G′, F, i)
Pairwise Nodes Connectivity Comparison
Node Similarity, Path-based Centrality and Link Appearance Measures
Weighted Link Scoring Computation
(u,v) ∈ E
Node-Dependent Metric
J_u,v = Jaccard(u,v,G)
Link Appearance (LA) Metric
L_u,v = LA(u,v,F)
ξ_u,v = (1- γ)·J_u,v + γ·L_u,v
(u,v) ∈ E′
(u,v) ∉ E
Node-Dependent Metric
J_u,v = Jaccard(u,v,G)
Link Appearance Metric
K_u,v = Katz(u,v,G,β)
ξ_u,v = (1- γ′)·J_u,v + γ′·K_u,v
Insert Link Score
S_i,j(u,v) = [class ξ_u,v u v]
Link Score Table: S
Si
Insert Si
Component 3: Algorithm 3
Link Anomaly Detection Algorithm(Θ, G, G′, Si, i)
Pairwise Nodes Connectivity and Score Comparison
Θ = {θ_build, θ_keep, θ_tear}
S_i,j(u,v) = [class ξ_u,v u v]
Link Anomaly Table: A
G=(V,E)
(u,v)∈E
u,v
if (u,v)∈E′ & ,ξ_u,v < θ_keep
else if (u,v)∉E′ & ,ξ_u,v > θ_tear
Insert Anomalous Link
G′=(V,E′)
(u,v)∈E′
u,v
if (u,v)∉E & ,ξ_u,v) < θ_build
A_i,j(u,v) = [class_true/false ξ_u,v u v]
Ai
Insert Ai

Figure 1. Data Flow Diagram

Algorithm 1: Sliding Time Window Algorithm($\tau, \omega, \omega', D, i$)

```
Input:
$\tau$ — time zone size
$\omega$ — prior window size
$\omega'$ — current window size
where $\omega \geq \omega'$
$D$ — raw network flow data: spatio-temporal data
$i$ — loop counter
Output:
$F$ — $i^{th}$ prior filtered data: spatio-temporal data
$F'$ — $i^{th}$ current filtered data: spatio-temporal data
$G = \langle V, E \rangle$ — $i^{th}$ prior graph: spatial data
$G' = \langle V, E' \rangle$ — $i^{th}$ current graph: spatial data
1  begin
2      foreach $timestamp \in ((i-1) \times \tau, (i-1) \times \tau + \omega]$ do
3          $D_i \leftarrow D(timestamp\ \ u\ ;\ v\ \ etc.)$
4          $F \leftarrow D_i(timestamp\ \ u\ ;\ v)$
5          $G(u, v) \leftarrow u, v$
6      end
7      foreach $timestamp' \in (i \times \tau + \omega - \omega', i \times \tau + \omega]$ do
8          $D'_i \leftarrow D(timestamp'\ \ u\ ;\ v\ \ etc.)$
9          $F' \leftarrow D_i(timestamp'\ \ u\ \ v)$
10         $G'(u, v) \leftarrow u, v$
11     end
12     return $F, G, G'$
13 end
```

Algorithm 2: Link Scoring Algorithm($\beta, \gamma, \gamma', G, G', F, i$)

```
Input:
$\beta$ — attenuation parameter
$\gamma, \gamma'$ — link scoring weights
$G = \langle V, E \rangle$ — $i^{th}$ prior graph
$G' = \langle V, E' \rangle$ — $i^{th}$ current graph
$F$ — $i^{th}$ prior filtered data
$i$ — loop counter
Output:
$S_i$ — $i^{th}$ link score table
$S_{i,j(u,v)} = [class \ \ \xi_{u,v} \ \ u \ \ v]$ — $j^{th}$ row vector of $S_i$
begin
    initialize $S_i$
    foreach $((u,v) \notin E) \wedge ((u,v) \in E')$ do
        $J_{u,v} \leftarrow \text{Jaccard}(u, v, G)$
        $K_{u,v} \leftarrow \text{Katz}(u, v, G, \beta)$
        $\xi_{u,v} \leftarrow (1 - \gamma') \cdot J_{u,v} + \gamma' \cdot K_{u,v}$
        $S_{i,j(u,v)} \leftarrow [build \ \ \xi_{u,v} \ \ u \ \ v]$
    end
    foreach $((u,v) \in E) \wedge ((u,v) \in E')$ do
        $J_{u,v} \leftarrow \text{Jaccard}(u, v, G)$
        $L_{u,v} \leftarrow \text{LinkAppearance}(u, v, F)$
        $\xi_{u,v} \leftarrow (1 - \gamma) \cdot J_{u,v} + \gamma \cdot L_{u,v}$
        $S_{i,j(u,v)} \leftarrow [keep \ \ \xi_{u,v} \ \ u \ \ v]$
    end
    foreach $((u,v) \in E) \wedge ((u,v) \notin E')$ do
        $J_{u,v} \leftarrow \text{Jaccard}(u, v, G)$
        $L_{u,v} \leftarrow \text{LinkAppearance}(u, v, F)$
        $\xi_{u,v} \leftarrow (1 - \gamma) \cdot J_{u,v} + \gamma \cdot L_{u,v}$
        $S_{i,j(u,v)} \leftarrow [tear \ \ \xi_{u,v} \ \ u \ \ v]$
    end
    return $S_i$
end
```

Algorithm 3: Link anomaly detection Algorithm(Θ, G, G', S_i, i)

```
Input:
$\Theta = \{\theta_{build}, \theta_{Linkclass!keep}, \theta_{tear}\}$ — thresholds
$G = \langle V, E \rangle$ — $i^{th}$ prior graph
$G' = \langle V', E' \rangle$ — $i^{th}$ current graph
$i$ — loop counter
$S_i$ — $i^{th}$ link score table
$S_{i,j(u,v)} = [class\ \ \xi_{u,v}\ \ u\ \ v]$ — $j^{th}$ row vector of $S_i$
Output:
$A_i$ — $i^{th}$ link anomaly table
$A_{i,j(u,v)} = [class_{true \vee false}\ \ \xi_{u,v}\ \ u\ \ v]$ — $j^{th}$ row vector of $A_i$
1  begin
2      initialize $A_i$
3      foreach $((u,v) \notin E) \wedge ((u,v) \in E')$ do
4          if $(\xi_{u,v} \leq \theta_{build})$ then
5          $A_{i,j(u,v)} \leftarrow [build_{true}\ \ \xi_{u,v}\ \ u\ \ v]$
6          else
7          $A_{i,j(u,v)} \leftarrow [build_{false}\ \ \xi_{u,v}\ \ u\ \ v]$
8      end
9      foreach $((u,v) \in E) \wedge ((u,v) \in E')$ do
10         if $(\xi_{u,v} \leq \theta_{Linkclass!keep})$ then
11         $A_{i,j(u,v)} \leftarrow [keep_{true}\ \ \xi_{u,v}\ \ u\ \ v]$
12         else
13         $A_{i,j(u,v)} \leftarrow [keep_{false}\ \ \xi_{u,v}\ \ u\ \ v]$
14     end
15     foreach $((u,v) \in E) \wedge ((u,v) \notin E')$ do
16         if $(\xi_{u,v} \geq \theta_{tear})$ then
17         $A_{i,j(u,v)} \leftarrow [tear_{true}\ \ \xi_{u,v}\ \ u\ \ v]$
18         else
19         $A_{i,j(u,v)} \leftarrow [tear_{false}\ \ \xi_{u,v}\ \ u\ \ v]$
20     end
21     return $A_i$
22 end
```

The flow-based programming procedures for the link anomaly detection are as follows:

1. Start running the program.
2. Input raw network traffic data D, time zone size τ, sliding time window size ω, link scoring weights γ and γ', an attenuation parameter β for Katz index, and a link anomaly detection threshold set $\Theta = \{\theta_{build}, \theta_{Linkclass!keep}, \theta_{tear}\}$.
3. Send input data sets to appropriate algorithmic components.
4. Initialize loop counter ($i = 1$).
5. Run *Component 1: Sliding Time Window Algorithm.* The network traffic data is parsed to get the prior window data (i.e., D_i, F, and G), and the current window data (i.e., D_i', F', and G') according to prior and current sliding window intervals. The i^{th} prior filtered data F, and prior and current graphs G and G' are sent to component 2 and/or component 3.
6. Run *Component 2: Link Scoring Algorithm.* The i^{th} link score table S_i is computed using F, G and G', and the results of (i) pairwise nodes connectivity statuses, (ii) node-dependent, path-dependent and link appearance measures, and (iii) combined weighted measures S_i are sent to components 3 and appended to S.
7. Run *Component 3: Link Anomaly Detection Algorithm.* The i^{th} link anomaly detection table A_i is determined by G, G', S_i and threshold set $\Theta = \{\theta_{build}, \theta_{Linkclass!keep}, \theta_{tear}\}$. The link anomaly detection is based on the results of (i) pairwise nodes connectivity statuses, and (ii) link score and thresholds comparisons. A_i is appended to A.
8. Increment loop counter i by one.
9. Run the above three components until the end of the network traffic data or a user stops the program.

An alternative method to create A is to extract A from S with subset pointers using the fact that A can be a subset of S ($A \subseteq S$) after classifying anomalies. In this case, S and subset pointers are stored in a database to further analyze link anomalies.

3.2.2. Sliding Time Window Algorithm

In our proposed DLAD framework (see Figure 1), the raw network traffic data D is processed to form two types of sliding time window based processed data, namely the prior window data and the current window data for unsupervised learning. The sliding time window algorithm (see Algorithm 1) controls the raw network traffic data to produce the i^{th} processed window data with paired window sizes ω and ω' and time zone size τ in the i^{th} sliding time window data processing. The prior window data consist of the i^{th} prior traffic data D_i, prior filtered data F and prior graph $(V, E) \in G$, and the current window data consist of current traffic data D_i', current filtered data F' and current graph $(V', E') \in G'$. The filtered data contain the spatial-temporal information (i,e, nodes, edges and times), and the graphs contain the spatial information (i.e., nodes and edges). The prior and current graphs are used to provide the conditional expressions for scoring and detecting anomalous links. In addition to the prior and current graphs, the prior filtered data is also used to weight the links to measure

Table 1: Link Classes and Mathematical Notations

Link Class	Mathematical Expression	Threshold
build	$((u,v) \notin E) \wedge ((u,v) \in E')$	θ_{build}
keep	$((u,v) \in E) \wedge ((u,v) \in E')$	$\theta_{Linkclass!keep}$
tear	$((u,v) \in E) \wedge ((u,v) \notin E')$	θ_{tear}
never linked	$((u,v) \notin E) \wedge ((u,v) \notin E')$	*N/A*

the temporal link appearance. Algorithm 1 shows the data representations of sliding time window data processing. With pointers, dynamic data structures of sliding time widow based processed data can be efficiently constructed or updated.

There are four classes of a link status for the paired sliding time window based link scoring and link anomaly detection algorithms, as follows:

1. *build* $((u,v) \notin E) \wedge ((u,v) \in E')$:
   A pair of nodes $u, v \in V$ was not connected in the prior graph, but the pair of nodes are built the new link in the current graph.
2. *keep* $((u,v) \in E) \wedge ((u,v) \in E')$:
   A pair of nodes $u, v \in V$ was connected in the prior graph and the pair of nodes are kept the link in the current graph.
3. *tear* $((u,v) \in E) \wedge ((u,v) \notin E')$:
   A pair of nodes $u, v \in V$ was connected in the prior graph, but the pair of nodes are torn down the link in the current graph.
4. *never link* $((u,v) \notin E) \wedge ((u,v) \notin E'))$:
   A pair of nodes $u, v \in V$ was not connected in both prior and current graphs. *never link* status is excluded for our proposed link anomaly detection.

For link anomaly detection, unsupervised learning is appropriate to learn the past histories of node-dependent, path dependent and link appearance behaviors and the prior and current link statuses. With the link statuses, the prior link scores are computed in Algorithm 2, and the anomalous prior links are detected in Algorithm 3.

3.2.3. Link Scoring Algorithm

The *component 2* in Figure 1 processes pairwise nodes connectivity comparison, network analytics measures and weighted link scoring computation, and produces the i^{th} link score table S_i (see Algorithm 2).

Two pairs of metrics are selected based on conditional expressions of link statuses as follows:

- *build* $((u,v) \notin E) \wedge ((u,v) \in E')$:

The *node-dependent metric* and the *path-dependent metric* are used to compute weighted link scores. In this case, node similarity provides limited information because the paired nodes $\{u, v\} \in V$ are not connected in the prior graph and these nodes are connected in the current graph. In the link anomaly detection, it is important to learn the new link path because the attack path length is a critical factor for the conventional anomaly detections. For example, longer path length is riskier than short path length in cyber attack data. Thus, in addition to the node similarity measure, the *path-dependent metric* is applied to determine whether a new link is anomalous.

- *keep* $((u, v) \in E) \wedge ((u, v) \in E')$ or *tear* $((u, v) \in E) \wedge ((u, v) \notin E')$:
  The *node-dependent metric* and the *link appearance metric* are used to compute weighted link scores. Link status of *keep* or *tear* may not be stable and the link status could change in a short time. Thus, in addition to node similarity and link centrality metrics, the *link appearance metric* is applied to provide temporal information of links, i.e., times and frequencies of link appearances in the prior filtered data, for temporal link analysis.

The computational complexity of the link scoring algorithm is polynomial time $O(|V|^3|E|)$. Since it is less efficient to compute scores for all pairwise nodes in the prior graph, only nodes that are actually connected in the current graph are used to compute the link scores, thus $|E|$ times. Relevant methods such as the Jaccard, Katz, and link appearance scores are applied for each link. Due to the cubic complexity time of the Katz measure which dominates the time complexity $(O(|V|^3))$, the overall complexity is $O(|V|^3|E|)$. If fast Katz [6] that runs at $O(|V|+|E|)$ is used, then computational complexity could be further reduced to $O(|E|^2 + |V||E|)$.

3.2.4. Link anomaly detection Algorithm

The link anomaly detection algorithm (see Algorithm 3) processes the i^{th} link anomaly table A_i using the prior and current graphs G, G', and the link score table S, with the user-defined threshold set Θ. Our link prediction on whether a link in a network graph is normal or anomalous is based on the conditional expressions of the link statuses in the prior and current graphs, prior link scores, and thresholds. The i^{th} link anomaly table A_i are appended to the link anomaly table A. The link anomaly detection algorithm's computational time complexity is $O(|E|)$.

The link score table S and the link anomaly table A are stored in the network security auditing database for further dynamic network analysis and evaluation of the proposed link scoring and detection algorithms.

4. RESULTS AND DISCUSSION

4.1. Case Study

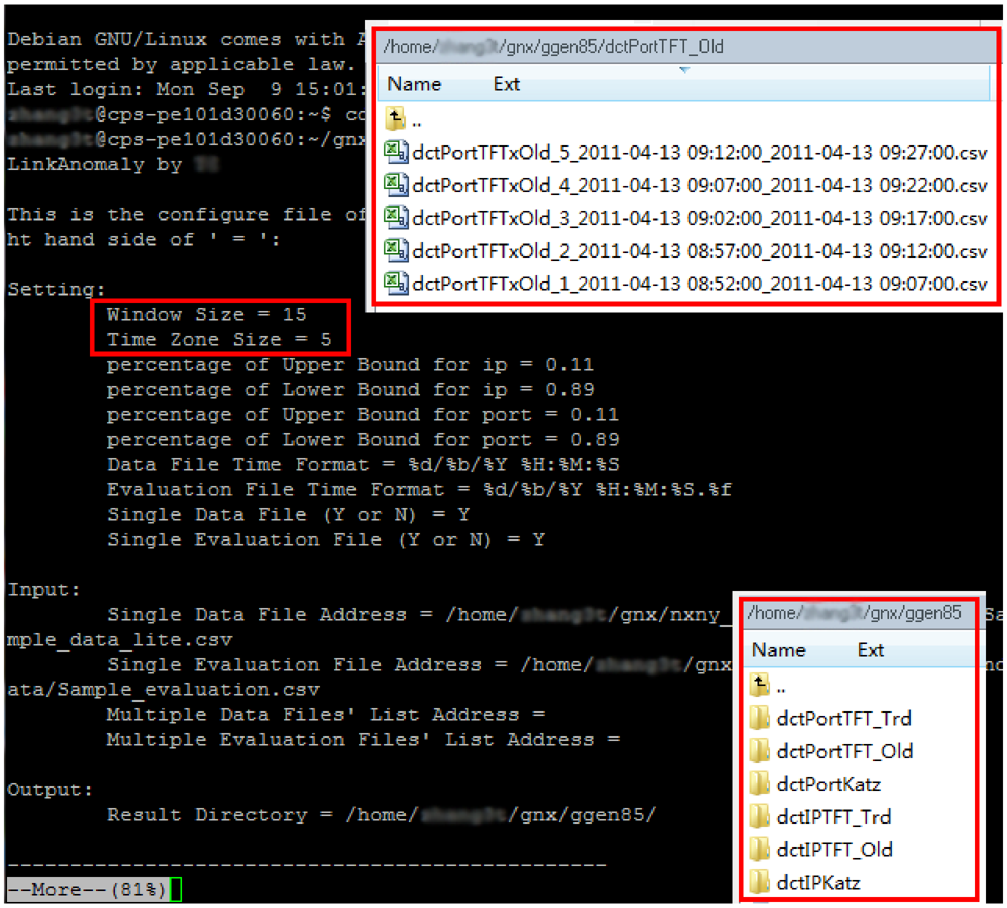
In this section, we conduct a case study using the VAST 2011 MC2 datasets that consist of three days of firewall and intrusion detection system (IDS) logs as core information data from the corporate network. These datasets are used to test and evaluate the proposed framework. The task is to identify some interesting events in the datasets for situational awareness in the corporate network. In this case study, the network flow data for log examination and security is investigated to show how the proposed framework may help to discover link anomalies.

4.1.1. Data Sets and Procedures

Network traffic data contain important information such as source (src) and destination (dst) Internet Protocol (IP) addresses, port numbers, timestamps, protocols, etc. We primarily focus on link anomaly detection using IP graphs. In an IP graph, each node represents one IP address and each edge represents a network connection between a source and destination IP pair. We can also construct IP-port graphs by introducing the port numbers, e.g., srcIP $\rightarrow$ srcPort $\rightarrow$ dstPort $\rightarrow$ dstIP, resulting in heterogeneous graphs. The log files are loaded into the dynamic link anomaly detection framework to construct two types of connectivity graphs with either IP or IP-port nodes. The two graphs give different insights on the networks with a tradeoff of granularity and complexity. The proposed framework employs flow-based programming to execute components using the paired window data sets, parameters, and thresholds for the link anomaly detection illustrated in 3.2.1.

4.1.2. Sliding Time Window Size Selection

For this case study, let the prior and current sliding time window sizes be the same, that is, $\omega = \omega'$. Suppose that the sliding window size $\omega = \omega'$ is 15 minutes and the time zone τ is 5 minutes (see Figure 2). With these parameters, the current window data are captured based on the raw network flow data in the current 15-minute window together with the overlapping prior 15-minute window (see Figure 1). The sliding time window controller will move the current and prior windows forward 5 minutes for capturing network flow data in the next iteration. The selections of the window size ω and time zone size τ are an important factor to generate suitable link scores for the link anomaly detection. Figures 3 and 4 show a comparison between the link appearance scores generated with a larger window size and a smaller window size. Figure 3 shows the scores of an IP-port graph starting at 2011-04-13 12:07:00 in a 1-minute window size in the VAST 2011 MC2 datasets. The extremely low link appearance scores are presented due to lack of connectivity records, making it hard to differentiate connections for the dynamic link anomaly detection. Figure 4 increases the graph size by enlarging the time window to 15 minutes. For this larger window, it is clear that larger scores make it easier to differentiate connections and are better for link anomaly detection. For other network flow data, a user should select other ω and τ because link scores can vary depending on the volume of network flow data.



```
Debian GNU/Linux comes with A
permitted by applicable law.
Last login: Mon Sep  9 15:01:
[redacted]@cps-pe101d30060:~$ cd
[redacted]@cps-pe101d30060:~/gnx
LinkAnomaly by [redacted]

This is the configure file of
ht hand side of ' = ':

Setting:
        Window Size = 15
        Time Zone Size = 5
        percentage of Upper Bound for ip = 0.11
        percentage of Lower Bound for ip = 0.89
        percentage of Upper Bound for port = 0.11
        percentage of Lower Bound for port = 0.89
        Data File Time Format = %d/%b/%Y %H:%M:%S
        Evaluation File Time Format = %d/%b/%Y %H:%M:%S.%f
        Single Data File (Y or N) = Y
        Single Evaluation File (Y or N) = Y

Input:
        Single Data File Address = /home/[redacted]/gnx/nxny_
mple_data_lite.csv
        Single Evaluation File Address = /home/[redacted]/gnx
ata/Sample_evaluation.csv
        Multiple Data Files' List Address =
        Multiple Evaluation Files' List Address =

Output:
        Result Directory = /home/[redacted]/gnx/ggen85/

------------------------------------------------
--More--(81%)
```

```
/home/[redacted]/gnx/ggen85/dctPortTFT_Old
Name        Ext
..
dctPortTFTxOld_5_2011-04-13 09:12:00_2011-04-13 09:27:00.csv
dctPortTFTxOld_4_2011-04-13 09:07:00_2011-04-13 09:22:00.csv
dctPortTFTxOld_3_2011-04-13 09:02:00_2011-04-13 09:17:00.csv
dctPortTFTxOld_2_2011-04-13 08:57:00_2011-04-13 09:12:00.csv
dctPortTFTxOld_1_2011-04-13 08:52:00_2011-04-13 09:07:00.csv
```

```
/home/[redacted]/gnx/ggen85
Name        Ext
..
dctPortTFT_Trd
dctPortTFT_Old
dctPortKatz
dctIPTFT_Trd
dctIPTFT_Old
dctIPKatz
```

Figure 2. Parameter Setting

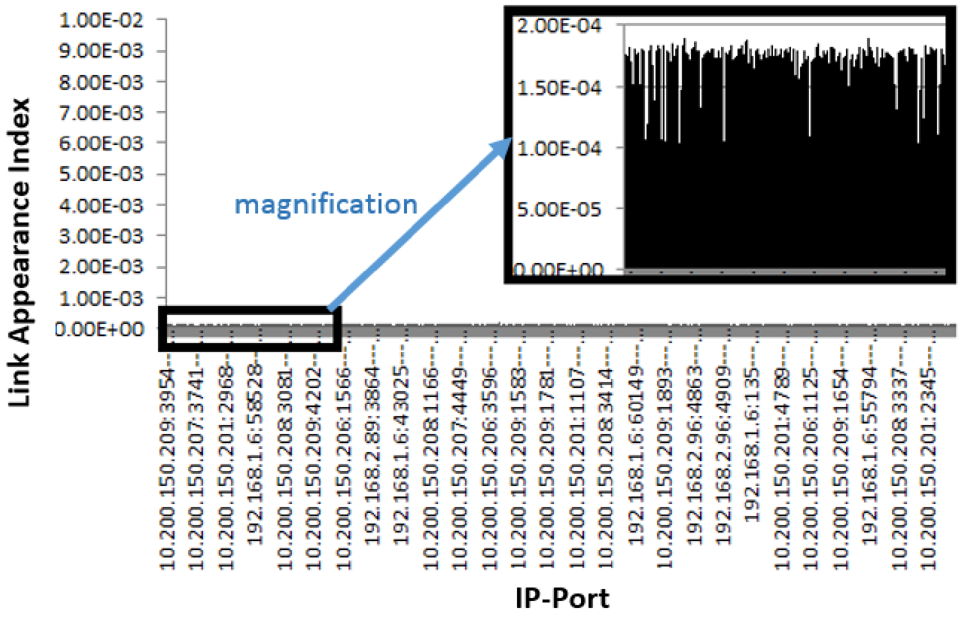


Figure 3. Link Appearance Indexes of an IP-Port Graph $\omega = 1min$

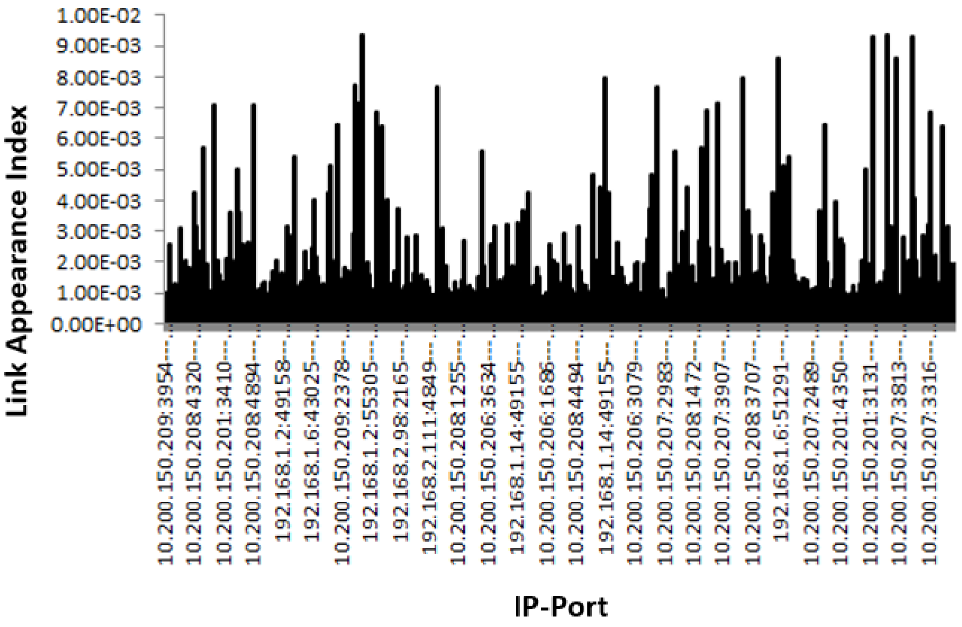
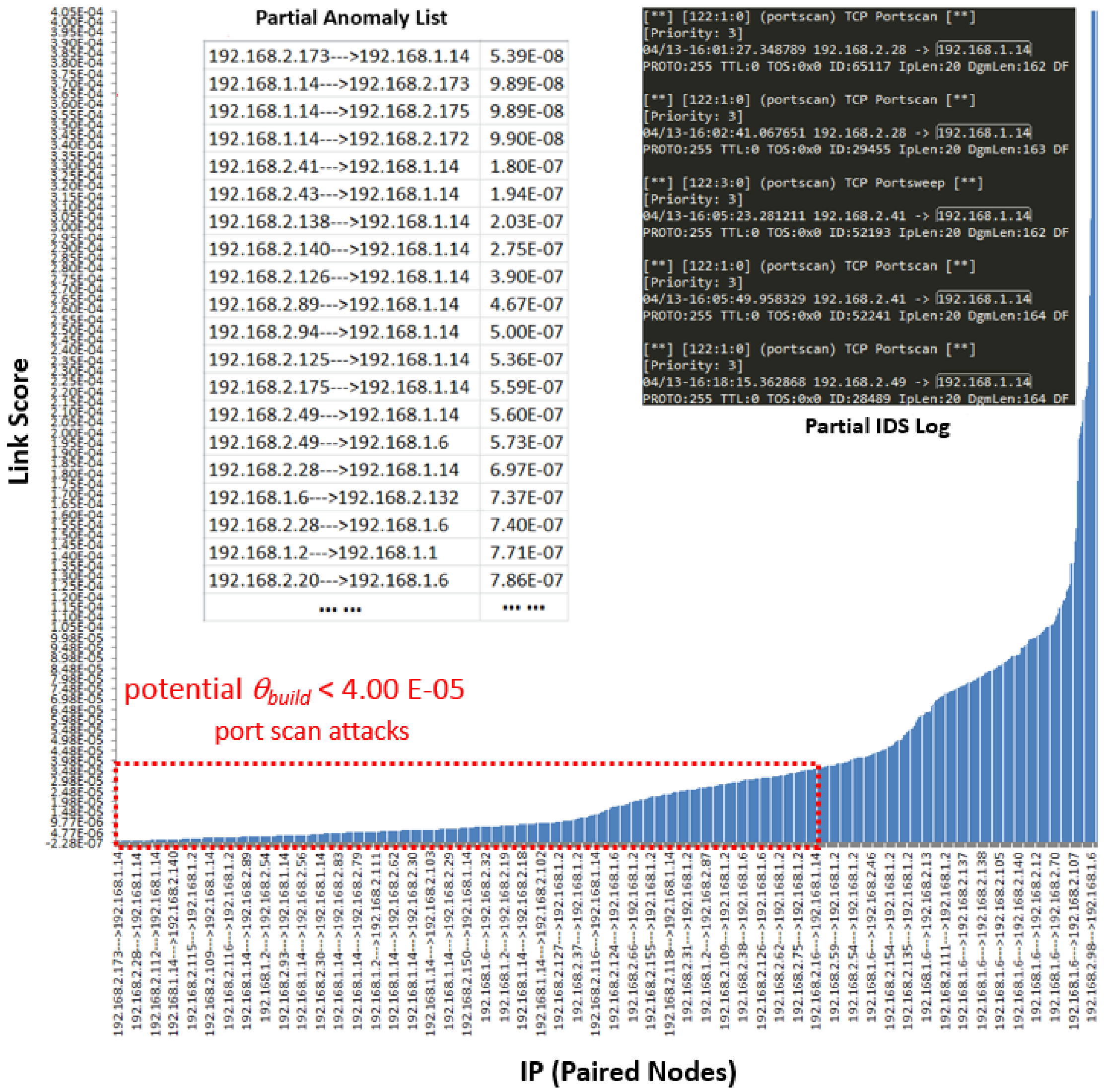


Figure 4. Link Appearance Indexes of an IP-Port Graph $\omega = 15min$

4.1.3. Thresholds for Cyber Attack Detection

The link score table S and the link anomaly table A are automatically generated. A user may further analyze the suspicious links in a specific time range. For example, Figure 5 partially shows a link score table that is ranked on day 1 (2011-4-13). Due to the space limitation on the horizontal axis, only a few paired IP addresses and their link scores are displayed. The node 192.168.1.14 and its associated nodes were built new connections and their link scores are very low during port scan attacks. Figure 5 indicates that $\theta_{build} < 4.00E - 05$ is a potential threshold to predict port scan attacks or other attacks. A group of machines with IP addresses ranging from 192.168.2.11 to 192.168.2.138 connect to three particular machines with IP 192.168.1.2, 192.168.1.6, and 192.168.1.14. Figure 6 shows this group and highlighted paired nodes that indicate confirmed port scan activities from the IDS log. In addition, workstations 192.168.2.171-175 are also the sources for many port scans to other hosts in the subnet. These link anomalies are confirmed as compromised machines which are starting to conduct port scanning and Denial of Service (DoS) attacks in the IDS log. Turning to the other side of the connection score distribution, Figure 7 presents the DoS attack related datasets and $\theta_{tear} > 0.1$ as a potential threshold to predict the DoS attacks. There was an attempted DoS attack against the corporate web server 172.20.1.15 for links with 10.200.150.201, 206-9 on day 1 (2011-4-13). As a result of DoS attacks, the corporate web server 172.20.1.15 was shut down. Subsequently all links that are connected to node 172.20.1.15 were torn down for a short time and yielded high link scores. While these examples in the case study are results of cyber attacks, the proposed framework could be useful in detecting any anomalous connections due to hardware failures or misconfigurations. Therefore, it may also benefit other network management tasks such as troubleshooting and



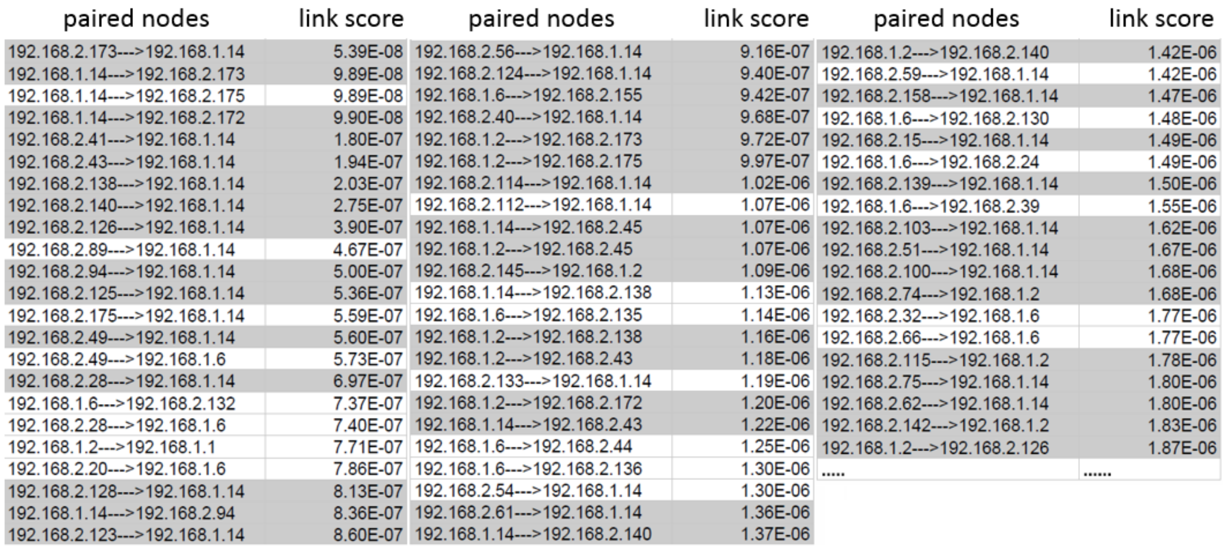
Partial Anomaly List

192.168.2.173--->192.168.1.14	5.39E-08
192.168.1.14--->192.168.2.173	9.89E-08
192.168.1.14--->192.168.2.175	9.89E-08
192.168.1.14--->192.168.2.172	9.90E-08
192.168.2.41--->192.168.1.14	1.80E-07
192.168.2.43--->192.168.1.14	1.94E-07
192.168.2.138--->192.168.1.14	2.03E-07
192.168.2.140--->192.168.1.14	2.75E-07
192.168.2.126--->192.168.1.14	3.90E-07
192.168.2.89--->192.168.1.14	4.67E-07
192.168.2.94--->192.168.1.14	5.00E-07
192.168.2.125--->192.168.1.14	5.36E-07
192.168.2.175--->192.168.1.14	5.59E-07
192.168.2.49--->192.168.1.14	5.60E-07
192.168.2.49--->192.168.1.6	5.73E-07
192.168.2.28--->192.168.1.14	6.97E-07
192.168.1.6--->192.168.2.132	7.37E-07
192.168.2.28--->192.168.1.6	7.40E-07
192.168.1.2--->192.168.1.1	7.71E-07
192.168.2.20--->192.168.1.6	7.86E-07
••• •••	••• •••

```
[**] [122:1:0] (portscan) TCP Portscan [**]
[Priority: 3]
04/13-16:01:27.348789 192.168.2.28 -> 192.168.1.14
PROTO:255 TTL:0 TOS:0x0 ID:65117 IpLen:20 DgmLen:162 DF

[**] [122:1:0] (portscan) TCP Portscan [**]
[Priority: 3]
04/13-16:02:41.067651 192.168.2.28 -> 192.168.1.14
PROTO:255 TTL:0 TOS:0x0 ID:29455 IpLen:20 DgmLen:163 DF

[**] [122:3:0] (portscan) TCP Portsweep [**]
[Priority: 3]
04/13-16:05:23.281211 192.168.2.41 -> 192.168.1.14
PROTO:255 TTL:0 TOS:0x0 ID:52193 IpLen:20 DgmLen:162 DF

[**] [122:1:0] (portscan) TCP Portscan [**]
[Priority: 3]
04/13-16:05:49.958329 192.168.2.41 -> 192.168.1.14
PROTO:255 TTL:0 TOS:0x0 ID:52241 IpLen:20 DgmLen:164 DF

[**] [122:1:0] (portscan) TCP Portscan [**]
[Priority: 3]
04/13-16:18:15.362868 192.168.2.49 -> 192.168.1.14
PROTO:255 TTL:0 TOS:0x0 ID:28489 IpLen:20 DgmLen:164 DF
```

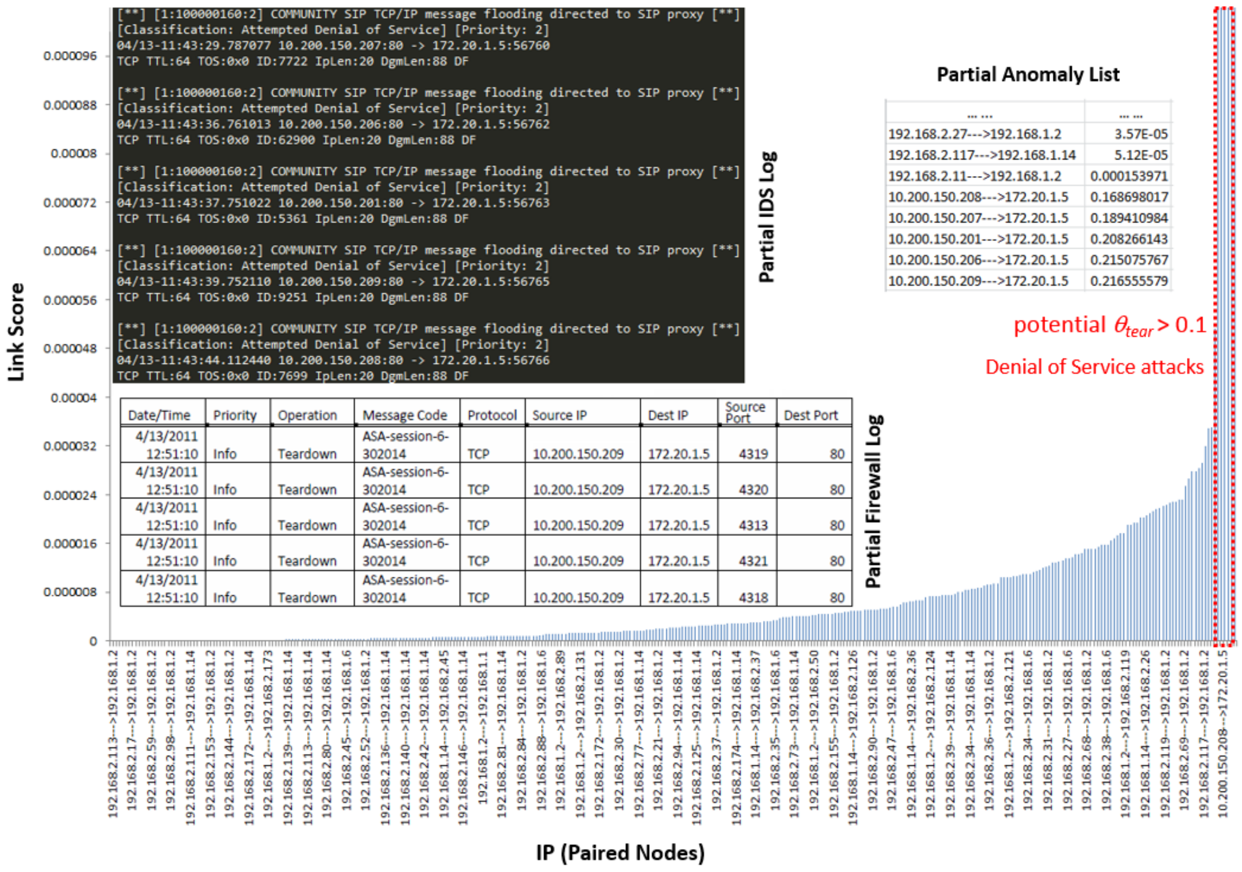
Partial IDS Log

Figure 5. Link Score Distribution of an IP Graph and θ_{build}



paired nodes	link score	paired nodes	link score	paired nodes	link score
192.168.2.173--->192.168.1.14	5.39E-08	192.168.2.56--->192.168.1.14	9.16E-07	192.168.1.2--->192.168.2.140	1.42E-06
192.168.1.14--->192.168.2.173	9.89E-08	192.168.2.124--->192.168.1.14	9.40E-07	192.168.2.59--->192.168.1.14	1.42E-06
192.168.1.14--->192.168.2.175	9.89E-08	192.168.1.6--->192.168.2.155	9.42E-07	192.168.2.158--->192.168.1.14	1.47E-06
192.168.1.14--->192.168.2.172	9.90E-08	192.168.2.40--->192.168.1.14	9.68E-07	192.168.1.6--->192.168.2.130	1.48E-06
192.168.2.41--->192.168.1.14	1.80E-07	192.168.1.2--->192.168.2.173	9.72E-07	192.168.2.15--->192.168.1.14	1.49E-06
192.168.2.43--->192.168.1.14	1.94E-07	192.168.1.2--->192.168.2.175	9.97E-07	192.168.1.6--->192.168.2.24	1.49E-06
192.168.2.138--->192.168.1.14	2.03E-07	192.168.2.114--->192.168.1.14	1.02E-06	192.168.2.139--->192.168.1.14	1.50E-06
192.168.2.140--->192.168.1.14	2.75E-07	192.168.2.112--->192.168.1.14	1.07E-06	192.168.1.6--->192.168.2.39	1.55E-06
192.168.2.126--->192.168.1.14	3.90E-07	192.168.1.14--->192.168.2.45	1.07E-06	192.168.2.103--->192.168.1.14	1.62E-06
192.168.2.89--->192.168.1.14	4.67E-07	192.168.1.2--->192.168.2.45	1.07E-06	192.168.2.51--->192.168.1.14	1.67E-06
192.168.2.94--->192.168.1.14	5.00E-07	192.168.2.145--->192.168.1.2	1.09E-06	192.168.2.100--->192.168.1.14	1.68E-06
192.168.2.125--->192.168.1.14	5.36E-07	192.168.1.14--->192.168.2.138	1.13E-06	192.168.2.74--->192.168.1.2	1.68E-06
192.168.2.175--->192.168.1.14	5.59E-07	192.168.1.6--->192.168.2.135	1.14E-06	192.168.2.32--->192.168.1.6	1.77E-06
192.168.2.49--->192.168.1.14	5.60E-07	192.168.1.2--->192.168.2.138	1.16E-06	192.168.2.66--->192.168.1.6	1.77E-06
192.168.2.49--->192.168.1.6	5.73E-07	192.168.1.2--->192.168.2.43	1.18E-06	192.168.2.115--->192.168.1.2	1.78E-06
192.168.2.28--->192.168.1.14	6.97E-07	192.168.2.133--->192.168.1.14	1.19E-06	192.168.2.75--->192.168.1.14	1.80E-06
192.168.1.6--->192.168.2.132	7.37E-07	192.168.1.2--->192.168.2.172	1.20E-06	192.168.2.62--->192.168.1.14	1.80E-06
192.168.2.28--->192.168.1.6	7.40E-07	192.168.1.14--->192.168.2.43	1.22E-06	192.168.2.142--->192.168.1.2	1.83E-06
192.168.1.2--->192.168.1.1	7.71E-07	192.168.1.6--->192.168.2.44	1.25E-06	192.168.1.2--->192.168.2.126	1.87E-06
192.168.2.20--->192.168.1.6	7.86E-07	192.168.1.6--->192.168.2.136	1.30E-06	......	......
192.168.2.128--->192.168.1.14	8.13E-07	192.168.2.54--->192.168.1.14	1.30E-06		
192.168.1.14--->192.168.2.94	8.36E-07	192.168.2.61--->192.168.1.14	1.36E-06		
192.168.2.123--->192.168.1.14	8.60E-07	192.168.1.14--->192.168.2.140	1.37E-06		

Figure 6. Link Score Tables



```
[**] [1:100000160:2] COMMUNITY SIP TCP/IP message flooding directed to SIP proxy [**]
[Classification: Attempted Denial of Service] [Priority: 2]
04/13-11:43:29.787077 10.200.150.207:80 -> 172.20.1.5:56760
TCP TTL:64 TOS:0x0 ID:7722 IpLen:20 DgmLen:88 DF

[**] [1:100000160:2] COMMUNITY SIP TCP/IP message flooding directed to SIP proxy [**]
[Classification: Attempted Denial of Service] [Priority: 2]
04/13-11:43:36.761013 10.200.150.206:80 -> 172.20.1.5:56762
TCP TTL:64 TOS:0x0 ID:62900 IpLen:20 DgmLen:88 DF

[**] [1:100000160:2] COMMUNITY SIP TCP/IP message flooding directed to SIP proxy [**]
[Classification: Attempted Denial of Service] [Priority: 2]
04/13-11:43:37.751022 10.200.150.201:80 -> 172.20.1.5:56763
TCP TTL:64 TOS:0x0 ID:5361 IpLen:20 DgmLen:88 DF

[**] [1:100000160:2] COMMUNITY SIP TCP/IP message flooding directed to SIP proxy [**]
[Classification: Attempted Denial of Service] [Priority: 2]
04/13-11:43:39.752110 10.200.150.209:80 -> 172.20.1.5:56765
TCP TTL:64 TOS:0x0 ID:9251 IpLen:20 DgmLen:88 DF

[**] [1:100000160:2] COMMUNITY SIP TCP/IP message flooding directed to SIP proxy [**]
[Classification: Attempted Denial of Service] [Priority: 2]
04/13-11:43:44.112440 10.200.150.208:80 -> 172.20.1.5:56766
TCP TTL:64 TOS:0x0 ID:7699 IpLen:20 DgmLen:88 DF
```

Date/Time	Priority	Operation	Message Code	Protocol	Source IP	Dest IP	Source Port	Dest Port
4/13/2011 12:51:10	Info	Teardown	ASA-session-6-302014	TCP	10.200.150.209	172.20.1.5	4319	80
4/13/2011 12:51:10	Info	Teardown	ASA-session-6-302014	TCP	10.200.150.209	172.20.1.5	4320	80
4/13/2011 12:51:10	Info	Teardown	ASA-session-6-302014	TCP	10.200.150.209	172.20.1.5	4313	80
4/13/2011 12:51:10	Info	Teardown	ASA-session-6-302014	TCP	10.200.150.209	172.20.1.5	4321	80
4/13/2011 12:51:10	Info	Teardown	ASA-session-6-302014	TCP	10.200.150.209	172.20.1.5	4318	80

... ...	... ...
192.168.2.27--->192.168.1.2	3.57E-05
192.168.2.117--->192.168.1.14	5.12E-05
192.168.2.11--->192.168.1.2	0.000153971
10.200.150.208--->172.20.1.5	0.168698017
10.200.150.207--->172.20.1.5	0.189410984
10.200.150.201--->172.20.1.5	0.208266143
10.200.150.206--->172.20.1.5	0.215075767
10.200.150.209--->172.20.1.5	0.216555579

Figure 7. Link Score Distribution of an IP Graph, Logs and Anomaly List

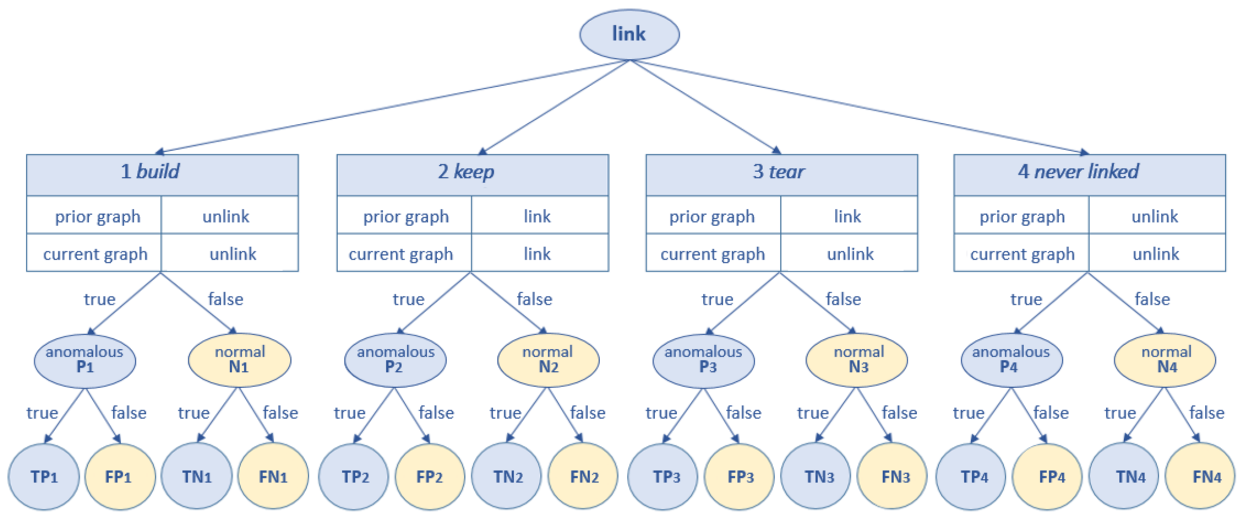


Figure 8. Multiclass Tree Representation of 4×4 Confusion Matrix

diagnosing.

4.2. Performance Evaluation

4.2.1. Link Anomaly Classifications

Figure 8 depicts multiclass tree representation of 4×4 confusion matrix that shows a decision-tree-like structure to illustrate the hierarchy of relationships regarding the 4 classes of a link status, the 8 predictive classes and the 16 instances of actual classification results. (Theoretically, for each link, there could be 16 classification results for 4 class-link anomaly detection.) The 4 classes of each link status are *build*, *keep*, *tear* and *never linked*, defined in Section 3.2.2. However, the *never linked* case is excluded in our current dynamic link anomaly analysis, and will instead be investigated in future research.

In predictive analytics, a confusion matrix is used to visualize the performance of a binary classifier. For each class of a link status, there are two predictive classes: anomalous (positive) or normal (negative). For each link anomaly prediction, the prediction can turn out to be either true or false, resulting in 4 instances such as true positive, false positive, true negative and false negative. These instances are defined (see Figure 8) as follows:

- True Positive (TP): Anomalous link (P) is correctly identified.
- False Positive (FP): Anomalous link (P) is incorrectly identified.
- True Negative (TN): Normal link (N) is correctly identified and rejected.
- False Negative (FN): Normal link (N) is incorrectly identified and rejected.

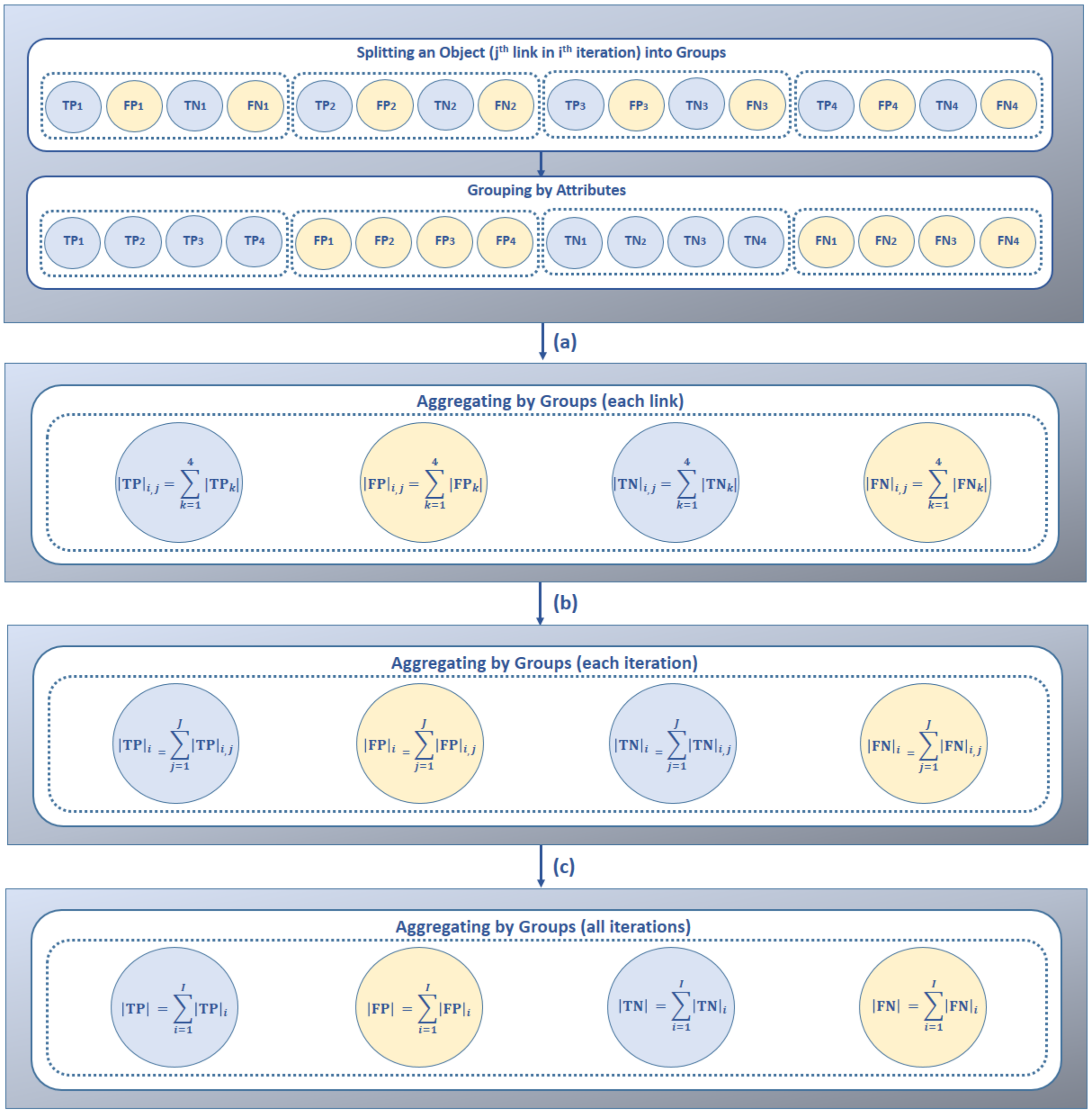

Splitting an Object (j^th link in i^th iteration) into Groups
TP1 FP1 TN1 FN1 TP2 FP2 TN2 FN2 TP3 FP3 TN3 FN3 TP4 FP4 TN4 FN4
Grouping by Attributes
TP1 TP2 TP3 TP4 FP1 FP2 FP3 FP4 TN1 TN2 TN3 TN4 FN1 FN2 FN3 FN4
(a)
Aggregating by Groups (each link)
$|TP|_{i,j} = \sum_{k=1}^{4} |TP_k|$
$|FP|_{i,j} = \sum_{k=1}^{4} |FP_k|$
$|TN|_{i,j} = \sum_{k=1}^{4} |TN_k|$
$|FN|_{i,j} = \sum_{k=1}^{4} |FN_k|$
(b)
Aggregating by Groups (each iteration)
$|TP|_i = \sum_{j=1}^{J} |TP|_{i,j}$
$|FP|_i = \sum_{j=1}^{J} |FP|_{i,j}$
$|TN|_i = \sum_{j=1}^{J} |TN|_{i,j}$
$|FN|_i = \sum_{j=1}^{J} |FN|_{i,j}$
(c)
Aggregating by Groups (all iterations)
$|TP| = \sum_{i=1}^{I} |TP|_i$
$|FP| = \sum_{i=1}^{I} |FP|_i$
$|TN| = \sum_{i=1}^{I} |TN|_i$
$|FN| = \sum_{i=1}^{I} |FN|_i$

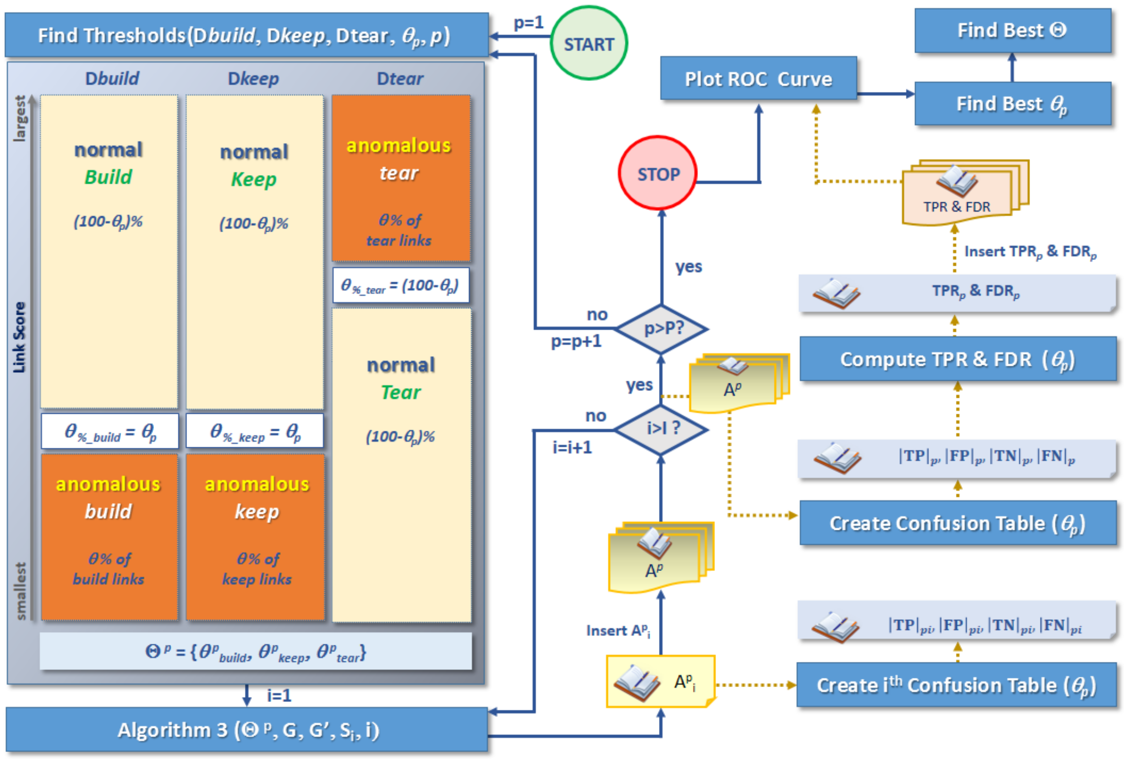
Figure 9. Hierarchical Aggregated Functions of Confusion Matrices

Table 2: Terminology and Derivations of Classification Functions from Confusion Matrix

Terminology		Classification Function	
Sensitivity	True Positive Rate	$TPR = \frac{TP}{P} = \frac{TP}{TP+FN}$	(5)
Specificity	True Negative Rate	$TNR = \frac{TN}{N} = \frac{TN}{TN+FP}$	(6)
1-Specificity	False Positive Rate	$FPR = \frac{FP}{N} = \frac{FP}{FP+TN}$	(7)
Precision	Positive Predictive Value	$PPV = \frac{TP}{TP+FP}$	(8)
Accuracy	Accuracy Rate	$ACC = \frac{TP+TN}{P+N} = \frac{TP+TN}{TP+FN+TN+FP}$	(9)
Error Rate	Recognition Rate	$ERR = \frac{FP+FN}{P+N} = \frac{FP+FN}{TP+FN+TN+FP}$	(10)

In this link anomaly research, four connection statuses (see 8 are identified and three of these connection statuses are used. These 3-class link statuses in the prior and current graphs are used to determine link scores and detect link anomalies. For overall performance evaluation, the hierarchical aggregated functions of confusion matrices regardless of link statuses are introduced to aggregate the classification results during the iterations or after completion of iterations (see Figure 9). Note that $|TP|_{i,j} + |FP|_{i,j} + |TN|_{i,j} + |FN|_{i,j} = 1$ for each link. The hierarchical aggregated functions enable one to convert multiclass classification to binary classification. From additional datasets such as IDS log, we are able to evaluate the link anomaly detection results via a confusion matrix.

Widely recognized classification functions (see Table 2) are selected to evaluate the proposed framework. These functions include sensitivity (true positive rate, TPR in Equation 5), specificity (true negative rate, TNR in Equation 6), 1-specificity (false positive rate, FPR in Equation 7), precision (positive predictive value, PPV in Equation 8), accuracy (accuracy rate, ACC in Equation 9), and error rate (recognition rate, ERR in Equation 10).



Find Thresholds(Dbuild, Dkeep, Dtear, θp, p)
p=1
START
Find Best Θ
Plot ROC Curve
Find Best θp
Dbuild
Dkeep
Dtear
largest
Link Score
smallest
normal Build (100-θp)%
normal Keep (100-θp)%
anomalous tear θ% of tear links
θ%_tear = (100-θp)
normal Tear (100-θp)%
θ%_build = θp
θ%_keep = θp
anomalous build θ% of build links
anomalous keep θ% of keep links
Θp = {θp_build, θp_keep, θp_tear}
STOP
TPR & FDR
Insert TPRp & FDRp
yes
no
p=p+1
p>P?
TPRp & FDRp
Compute TPR & FDR (θp)
yes
Ap
no
i=i+1
i>I ?
|TP|p, |FP|p, |TN|p, |FN|p
Create Confusion Table (θp)
Ap
Insert Api
|TP|pi, |FP|pi, |TN|pi, |FN|pi
Api
Create ith Confusion Table (θp)
i=1
Algorithm 3 (Θp, G, G', Si, i)

Figure 10. 3-Threshold Selection Procedure Using a Binary Classifier

4.2.2. Multi-Thresholds Selection

Figure 10 illustrates the thresholds selection procedures. S is a link score table and S_{build}, $S_{Linkclass!keep}$, S_{tear} are extracted from S. Each class link score table is individually ranked. A simple strategy can be used to find $\Theta = \{\theta_{build}, \theta_{Linkclass!keep}, \theta_{tear}\}$, and use Receiver Operating Characteristic (ROC) curve is used to determine the best Θ. Thresholds selection procedures are as follows:

1. Let $\psi^p \in \Psi$. ψ^p is the p^{th} percentage of link score sets where $p = 1, 2, \cdots, P$.
2. Select $\Theta^p = \{\theta^p_{build}, \theta^p_{Linkclass!keep}, \theta^p_{tear}\}$ by mapping the ψ^p % of S_{build}, S_{build} and S_{tear}.
3. Run Algorithm 3 with Θ^p.
4. Use the aggregated functions in Figure 9 to create p^{th} confusion matrix.
5. Compute TPR^p and FDR^p.
6. Plot ROC curve using all TPR^p and FDR^p, and determine the best ψ^p.
7. Map the best ψ^p to select $\Theta = \{\theta_{build}, \theta_{Linkclass!keep}, \theta_{tear}\}$.

4.2.3. Performance of Classification

The ROC curve is an effective method to select the best operating point for a binary classifier. We use the ROC curve to determine three thresholds to classify the three link classes. Estimation of multi-thresholds for link anomaly detection is challenging because of dynamic

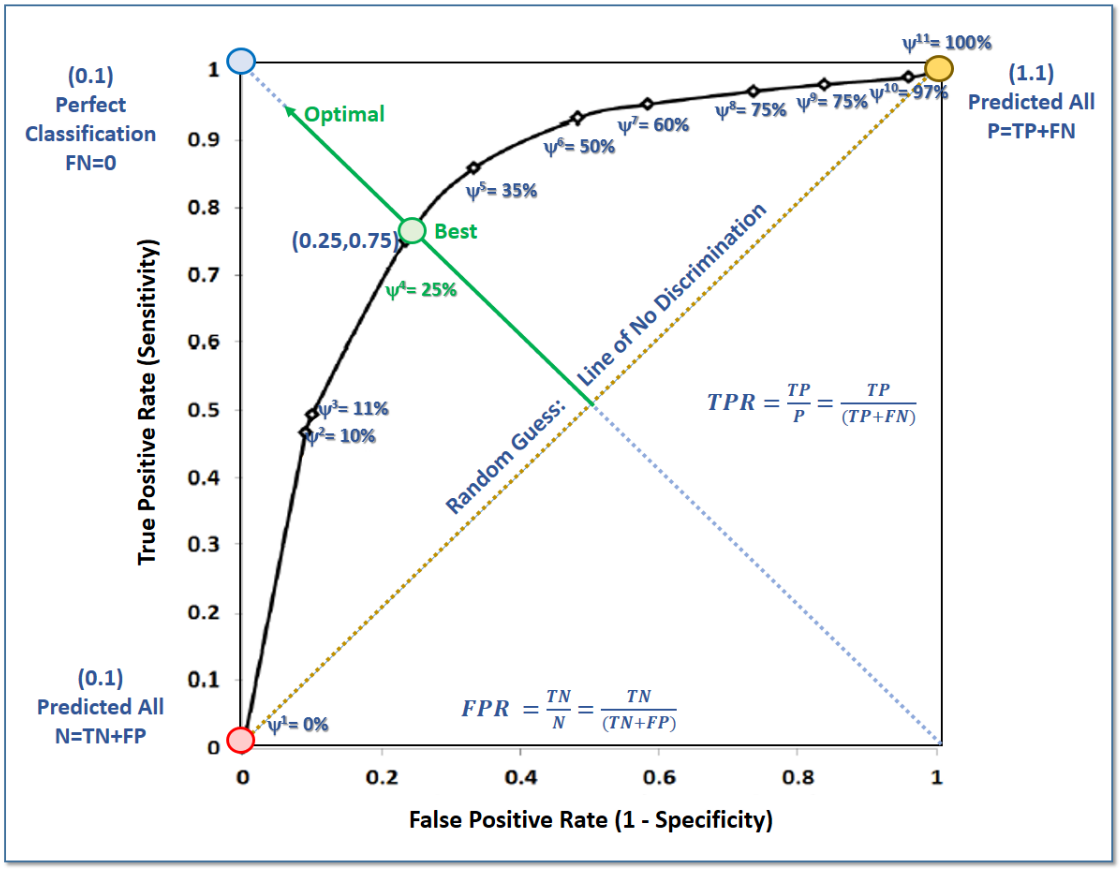


Figure 11. Alternative Comparison of Various Thresholds in ROC Curve

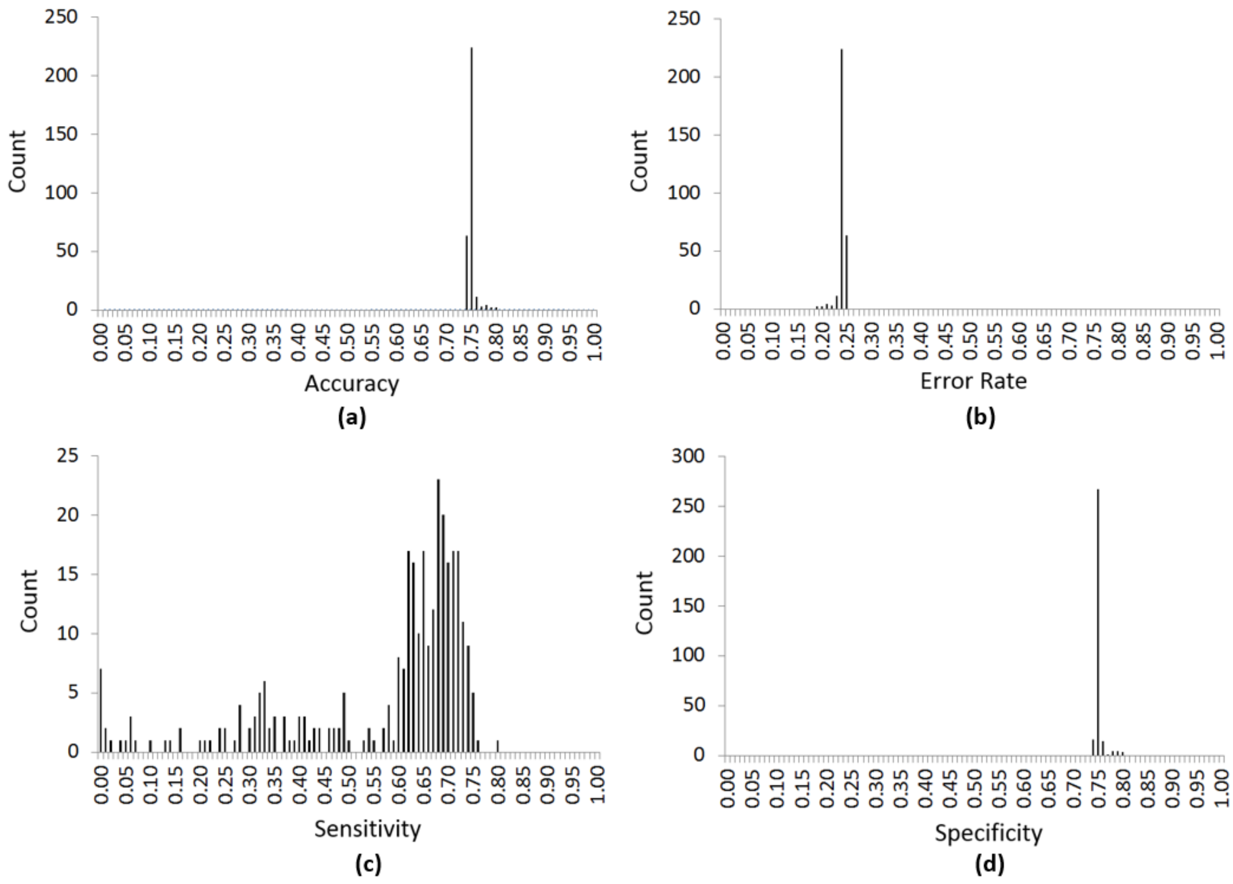


Figure 12. Performance Evaluation Results

link behaviors. Since VAST 2011 MC2 datasets were produced by simulating normal activity and attacks, we apply the simple method of multi-thresholds estimation, instead of using conventional multi-thresholds estimation techniques. The proposed approach is described in Section 4.2.2. Figure 11 presents the operating points (i.e., $\psi^p \in \Psi^p$) and the ROC curve indicates that $\psi^4 = 25\%$ is the best operating point for mapping the thresholds $\Theta^4 = \{\theta^4_{build}, \theta^4_{Linkclass!keep}, \theta^4_{tear}\}$ that can be used as $\Theta = \{\theta_{build}, \theta_{Linkclass!keep}, \theta_{tear}\}$. For the DLAA, $p \geq 5$ (i.e., $\psi^p \geq 25\%$) can be selected to analyze more suspicious links in the VAST 2011 MC2 datasets. This multi-thresholds selection method is inappropriate for real network traffic data, but it is sufficient to demonstrate the effectiveness of sliding time window based data stream mining.

After selecting the threshold set Θ, we perform 300 rounds of link anomaly detections with the sliding time window starting from the beginning of the VAST 2011 MC2 network flow data and moving towards the end of the data. For each time window, paired prior and current window data are updated. The prior window data are used for learning and the current window data are used for predicting link anomalies. The IDS log is used for verification purposes.

For each round of testing, accuracy, error rate, sensitivity and specificity are measured using the aggregated confusion matrix for overall performance of link anomaly predictions (see Figure 9). These measurements are used to evaluate the proposed framework. Figure 12 shows that the experimental results yielded roughly 75% overall accuracy, indicating the effectiveness of scoring and detecting the link anomaly using the proposed framework. The performance of the framework may be degraded unless parameters and thresholds are adaptable for the real dynamic link data. However, the framework is appropriate to filter and analyze anomalous links by adjusting threshold sets $\Theta$and parameters.

5. CONCLUSION

The DLAD framework of this report was developed to quantify the security levels of dynamic networks, and evaluated to show the effectiveness of data stream mining for the DNA. Through performance evaluation, we demonstrated that the proposed framework has a capability to construct effective knowledge structures by measuring the security levels of dynamic networks, and filtering anomalous or suspicious links from massive network traffic data. In addition, the proposed sliding time window based method produces useful processed data for data stream mining based generalized dynamic network analysis. This DLAA method needs to be further developed to incorporate statistical based data stream mining.

6. REFERENCES

[1] Akoglu, L., McGlohon, M., and Faloutsos, C., "OddBall: Spotting Anomalies in Weighted Graphs," *Proceedings of the 14th Pacific-Asia Conference on Knowledge Discovery and Data Mining*, PAKDD'10, Hyderabad, India, (2010).

[2] Almansoori, W., Gao, S., Jarada, T. N., Elsheikh, A. M., Murshed, A. N., Jida, J., Alhajj, R., and Rokne, J., "Link Prediction and Classification in Social Networks and its Application in Healthcare and Systems Biology," *Network Modeling Analysis in Health Informatics and Bioinformatics*, Vol. 1:1-2, pp. 27–36, 2012.

[3] Bonchi, F., Esfandiar, P., Gleich, D. F., Greif, C., and Lakshmanan, L. V., "Fast Matrix Computations for Pairwise and Columnwise Commute Times and Katz Scores," *Internet Mathematics*, Vol. 8:1-2, pp. 73–112, 2012.

[4] Chakrabarti, D. and Faloutsos, C., "Graph Mining: Laws, Generators, and Algorithms," *ACM Computing Surveys*, Vol. 38:1, pp. 1–69, 2006.

[5] Fire, M., Tenenboim, L., Lesser, O., Puzis, R., Rokach, L., and Elovici, Y., "Link Prediction in Social Networks Using Computationally Efficient Topological Features," *Proceedings of the 2011 IEEE International Conference on Privacy, Security, Risk and Trust and 2011 IEEE International Conference on Social Computing*, PASSAT/SocialCom'11, Boston, MA, USA, (2011).

[6] Foster, K. C., Muth, S. Q., Potterat, J. J., and Rothenberg, R. B., "A Faster Katz Status Score Algorithm," *Computational & Mathematical Organization Theory*, Vol. 7:4, pp. 275–285, 2001.

[7] García-Teodoro, P., Díaz-Verdejo, J., Maciá-Fernández, G., and Vázquez, E., "Anomaly-based Network Intrusion Detection: Techniques, Systems and Challenges," *Computers & Security*, Vol. 28, pp. 18–28, 2009.

[8] Getoor, L. and Diehl, C. P., "Link Mining: a Survey," *ACM SIGKDD Explorations Newsletter*, Vol. 7:2, pp. 3–12, 2005.

[9] Golub, G. and Van Loan, C., **Matrix Computations**, Johns Hopkins University Press, Baltimore, MD, USA, 2nd edition, 1989.

[10] Hasan, M. A., Chaoji, V., Salem, S., and Zaki, M., "Link Prediction using Supervised Learning," *Proceedings of the 6th SIAM International Conference on Data Mining, Workshop on Link Analysis, Counterterrorism and Security*, SDM'06, Bethesda, MD, USA, (2006).

[11] Huang, Z. and Zeng, D., "A Link Prediction Approach to Anomalous Email Detection," *Proceedings of the 2006 IEEE International Conference on Systems, Man, and Cybernetics*, SMC'06, Taipei, Taiwan, (2006).

[12] Katz, L., "A New Status Index Derived from Sociometric Analysis," *Journal Psychometrika*, Vol. 18:1, pp. 39–43, 1953.

[13] Lakhina, A., Crovella, M., and Diot, C., "Characterization of Network-wide Anomalies in Traffic Flows," *Proceedings of the 4th ACM SIGCOMM Conference on Internet Measurement*, IMC'04, Taormina, Sicily, Italy, (2004).

[14] Liao, Q. and Striegel, A., "Intelligent Network Management Using Graph Differential Anomaly Visualization," *The Proceedings of the 2012 IEEE Network Operations and Management Symposium*, NOMS'12, Maui, HI, USA, (2012).

[15] Liben-Nowell, D. and Kleinberg, J., "The Link Prediction Problem for Social Networks," *Proceedings of the 12th International Conference on Information and Knowledge Management*, CIKM'03, New Orleans, LA, USA, (2003).

[16] Liben-Nowell, D. and Kleinberg, J., "The Link-prediction Problem for Social Networks," *journal of the Association for Information Science and Technology*, Vol. 58:7, pp. 1019–1031, 2007.

[17] Lichtenwalter, R. N., Lussier, J. T., and Chawla, N. V., "New Perspectives and Methods in Link Prediction," *Proceedings of the 16th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD'10, Washington D.C.,USA, (2010).

[18] Lü, L. and Zhou, T., "Link Prediction in Complex Networks : A Survey," *Physica A: Statistical Mechanics and Its Applications*, Vol. 6:390, pp. 1150–1170, 2011.

[19] O'Madadhain, J., Hutchins, J., and Smyth, P., "Prediction and Ranking Algorithms for Event-Based Network Data," *ACM SIGKDD Explorations Newsletter*, Vol. 7:2, pp. 23–30, 2005.

[20] Potgieter, A., April, K., Cooke, R., and Osunmakinde, I., "Temporality in Link Prediction : Understanding Social Complexity," *Sprouts: Working Papers on Information Systems*, Vol. 7:9, pp. 1–25, 2007.

[21] Rattigan, M. J. and Jensen, D., "The Case For Anomalous Link Discovery," *ACM SIGKDD Explorations Newsletter*, Vol. 7:2, pp. 41–47, 2005.

[22] Sommer, R. and Paxson, V., "Outside the Closed World: On Using Machine Learning for Network Intrusion Detection," *Proceedings of the 2010 IEEE Symposium on Security and Privacy*, SP'10, Oakland, CA, USA, (2010).

[23] Sun, J., Qu, H., Chakrabarti, D., and Faloutsos, C., "Neighborhood Formation and Anomaly Detection in Bipartite Graphs," *Proceedings of the 2005 IEEE International Conference on Data Mining*, ICDM'05, Houston, TX, USA, (2005).

[24] Wan, X., Milios, E., Kalyaniwalla, N., and Janssen, J., "Link-based Anomaly Detection in Communication Networks," *Proceedings of the 2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, WI-IAT'08, Sydney, Australia, (2008).

Index

Algorithmic component, 1, 6, 11
  link anomaly detection, 1, 10, 11, 13
  link scoring, 1, 11, 12
  link scoring algorithm, 9
  sliding time window, 1, 11
  sliding time window algorithm, 8, 11
Anomaly detection, 2, 3
Attenuation parameter, 5, 11

Binary classifier, 19, 22
Bipartite graph, 2

Cardinality, 4
Centrality, 4–6, 13
Characteristics, 2, 3
Complex networks, 6
Complexity, 2, 6, 13
Conditional expression, 11, 12
Confusion matrix, 19, 22, 25
  false negative (FN), 19
  false positive (FP), 19
  true negative (TN), 19
  true positive (TP), 19
Connectivity, 3, 5, 14

Data flow diagram, 6
Data mining, 2
Data processing, 11, 12
Data representation, 12
Data stream mining, 1, 24, 25
Denial of Service (DoS), 2, 16
Dependability, 2
DLAD framework, 1
Dynamic data structure, 12
Dynamic link anomaly analysis (DLAA), 1, 25
Dynamic link anomaly detection (DLAD), 1, 3, 6, 11, 14
Dynamic network analysis (DNA), 1, 13, 25
Dynamic networks, 1, 25

Eigenvalues, 2
Evaluation metrics, 3

Filtering, 25
Firewall, 14
Flow-based programming, 1, 11, 14
Functionality, 2

Graph, 2, 4, 11, 14
Graph-theoretic approach, 2

Heterogeneous graph, 14

Intrusion detection system (IDS), 14

Jaccard index, 4, 6
Jaccard similarity coefficient, 4

Katz index, 5, 6, 11
Knowledge structures, 1, 3, 25

Link anomaly, 5, 6, 11–14, 19
Link anomaly table, 13
Link appearance, 12
Link appearance metric, 5, 13
Link class, 12, 22
  build, 12
  keep, 12, 13
  never link, 12
  tear, 12, 13
Link prediction algorithm, 2, 3
Link score table, 11–13

Machine learning, 2
Malicious activities, 3
Mapping, 24
Misconfiguration, 3
Multi-thresholds, 22, 24
Multiclass tree representation, 19
Multivariate time series, 2

Network analysis, 4
Network analytics metric, 1, 4
  link appearance metric, 5, 6
  node-dependent metric, 6
  path-dependent metric, 6
Network behavior anomaly detection (NBAD), 2

Network caching, 5
Network connections, 5
Network flow data, 14
Network science, 1
Network security, 1–3
Network security auditing, 13
Network topologies, 2–4
Network traffic data, 2, 11, 14, 24, 25
Node-dependent metric, 4, 13
Nonlinear weight function, 5

OddBall algorithm, 2
Outlierness, 2

Paired data, 1, 24
Paired sliding time windows, 12
Paired window sizes, 11
Path-dependent metric, 5, 13
Performance of a binary classification
    accuracy, 1, 24
    error rate, 1, 24
    sensitivity, 1, 24
    specificity, 1, 24
Polynomial time, 13
Power law density, 2
Precision, 1
Prediction, 1–3, 13
Predictions, 3
Processed data, 1, 11, 12, 25

Rank, 2
Receiver Operating Characteristic (ROC), 22

Security investigation, 3
Security levels, 1, 25
Security related events, 3
Similarity, 4, 6, 13
Situation awareness, 14
Sliding time window, 1
Sliding time window data processing, 11
Sliding time window size, 11, 14
Snapshot graph, 5
Social network analysis, 4
Social networks, 2
Spatial data, 1
Spatial information, 11
Spatio-temporal data, 1
Spatio-temporal link analysis, 1, 4
Static graph, 2
Subgraph, 2
Subspace, 2

Temporal dynamic, 4, 5
Temporal information, 13
Temporal link analysis, 13
Temporal locality, 4, 5
Temporality, 2
Threshold, 11, 22, 24
Time zone size, 11
Topological structure, 3, 4

Unsupervised learning, 1, 11, 12

Weighted link score, 13
Weighted mean, 5
Weights, 2, 5, 11